ARCHITEKTURA APLIKACJI SERWEROWEJ SYSTEMU TELEMEDYCZNEGO DO ZASTOSOWAŃ W MONITORINGU ŚRODOWISKOWYM PACJENTA

Wojciech Surtel, Marcin Maciejewski, Rafał Różalski
Politechnika Lubelska, Wydział Elektrotechniki i Informatyki, Instytut Elektroniki i Technik Informacyjnych

Streszczenie. *W celu poprawnego zaimplementowania systemu telemedycznego konieczne jest stworzenie i wdrożenie odpowiednio przygotowanej centralnej aplikacji serwerowej. Aplikacja taka została opisana w poniższym artykule. Powinna ona zapewniać odpowiednią funkcjonalność i udostępniać wymagany zakres danych zarówno dla pacjentów jak i operatorów i personelu medycznego. Konieczne jest wykorzystanie odpowiednio dostosowanego protokołu komunikacyjnego w celu jednoczesnego zapewnienia kompletności informacji i zminimalizowania obciążenia dla terminali mobilnych. Jednocześnie wymagane jest podzielenie aplikacji na odpowiednie warstwy w celu ułatwienia integracji w istniejących systemach i wdrażania. Konieczne jest również zapewnienie bezbłędnej komunikacji i wysokiego poziomu bezpieczeństwa podczas uzyskiwania dostępu do chronionych danych medycznych.*

Słowa kluczowe: telemedycyna, baza danych, aplikacja serwerowa, protokoły, medycyna

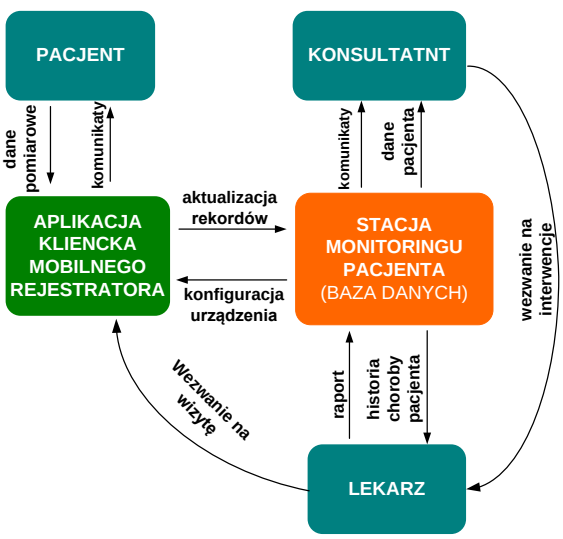
ARCHITECTURE OF A SERVER APPLICATION FOR USE IN ENVIRONMENTAL PATIENT MONITORING

Abstract. *Proper telemedical system implementation requires a central server application for storing and managing data and diagnostics messages. Such an application is described in the presented article below. It should provide sufficient functionality and allow for appropriate access privileges for different groups of users, including patients, operators and medical staff. A properly designed protocol must be used to simultaneously provide complete and safe information and minimize load on mobile terminals. At the same time it is necessary to divide the application into proper layers to be easily integrated into existing medical systems and make implementation easier. It is also of utmost importance to provide a high level of safety during access to protected, sensitive data.*

Keywords: telemedicine, database, application, protocol, medicine

Wstęp

Zastosowania telemedyczne są ściśle związane z rozwojem społeczeństwa informacyjnego i zakresu medycznej telemetyki. Systemy telemedyczne służące monitorowaniu wybranych funkcji życiowych pacjenta umożliwiają w sposób ciągły zdalny nadzór nad pacjentem. Wiele systemów telemedycznych jest mocno scentralizowanych, opierając się na wymianie danych między konkretną stacją roboczą (najczęściej komputerem pacjenta) a serwerem głównym. Zastosowanie urządzeń mobilnych pozwala na rozwijanie istniejących systemów scentralizowanych w kierunku systemów rozproszonych. Nowe technologie stosowane do budowy mobilnych rejestratorów funkcji życiowych pacjenta (stacji roboczych pacjenta) wymagają specjalizowanych rozwiązań w budowie aplikacji serwerowej oraz protokołów komunikacyjnych. W opisywanym rozwiązaniu aplikacja serwerowa wymienia się danymi z aplikacją mobilną poprzez dedykowany protokół bezpiecznej wymiany danych UWD. Na rysunku 1 przedstawiono model kontekstowy systemu.



Rys. 1. Model kontekstowy proponowanego systemu

W niniejszej pracy przedstawiono opis aplikacji serwerowej do zastosowania w stacji monitoringu pacjenta będącej centralnym elementem systemu telemedycznego dostosowanego do współpracy z mobilnymi rejestratorami funkcji życiowych [4].

1. Opis działania aplikacji serwerowej

System telemedyczny ma umożliwić pacjentom wykonywanie badań z wykorzystaniem dostępnych urządzeń, wysłać dane do lekarza w celu analizy i kontroli stanu zdrowia. Wykorzystując współczesne technologie takie jak Internet, użytkownik ma możliwość zdalnej diagnostyki oraz archiwizacji swoich badań, włączając w to interakcję grup użytkowników (lekarze, konsultanci). Wymaga to stworzenia odpowiedniego środowiska oraz infrastruktury, która zapewniłaby prawidłowe działanie takiego systemu.

W opisywanym projekcie aplikacja serwerowa ma jedno z bardziej kluczowych funkcji. Serwer jest zazwyczaj miejscem centralnym systemu, który udostępnia pewne zasoby innym komputerom lub pośredniczy w przekazywaniu danych pomiędzy urządzeniami. Współdzielenie zasobów odbywa się na zasadzie klient - serwer, gdzie w tym przypadku klientem będzie aplikacja mobilna z podłączonymi rejestratorami. Głównym zadaniem aplikacji serwerowej będzie komunikacja z urządzeniem mobilnym, gromadzenie danych oraz odpowiednia ich wizualizacja dla konkretnych grup użytkowników. Serwer ma za zadanie w sposób płynny umożliwić komunikację oraz odpowiednią interakcję pomiędzy poszczególnymi grupami użytkowników, dzięki czemu będzie możliwa natychmiastowa interwencja odpowiednich służb ratunkowych [2].

1.1. Wymagania funkcjonalne

Aplikację serwerową można podzielić na dwie zasadnicze części: serwer oraz aplikację www. Zadaniem serwera jest:

- realizacja sesji komunikacyjnej z aplikacją mobilną,
- gromadzenie danych z sesji,
- generowanie powiadomień dla użytkowników,

Natomiast aplikacja www ma umożliwić:

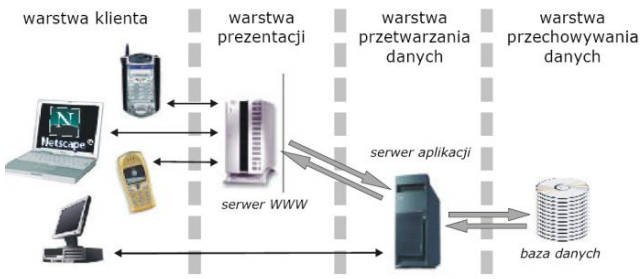
- zarządzanie kontami użytkowników,
- zarządzanie urządzeniami powiązanymi z użytkownikami,
- dostęp do danych gromadzonych z rejestratorów,
- dostęp do danych dla grup powiązanych z użytkownikiem (lekarz, konsultant).

1.2. Budowa aplikacji serwerowej

Aplikacja serwerowa została napisana w języku PHP, który od wersji 5 został wzbogacony o model obiektowy. Dzięki swojej prostocie PHP jest obecnie językiem najczęściej stosowanym

do generowania stron WWW. Skrypty PHP można w łatwy sposób zamieszczać w plikach takich jak HTML bądź XHTML. Lecz nie chodzi tutaj o generowanie stron z wykorzystaniem języka skryptowego lecz także o umożliwienie prostego mechanizmu, który jednocześnie zapewni komunikację z urządzeniem mobilnym. Do pomocy wykorzystano framework Symfony 2. Jest to rozbudowana biblioteka skupiająca w sobie wiele możliwości jakie dają obecnie platformy Java EE czy ASP .NET, które wykorzystują model obiektowy. Symfony 2 do renderowania widoków wykorzystuje silnik TWIG, który został także wykorzystany także do generowania odpowiedzi przez serwer, co w znacznym stopniu ułatwiło odpowiednie zarządzanie wiadomościami. Dodatkowo do komunikacji z bazą danych posłużono się wbudowaną biblioteką Doctrine ORM 2.0 umożliwiającą mapowanie obiektów PHP na relacyjne bazy danych.

Aplikacja serwerowa dzieli się na cztery funkcjonalne warstwy [1]. Model warstwicowy pomaga rozdzielić poszczególne moduły i przydzielić im odpowiednią rolę w systemie. Poniższy rysunek przedstawia zależności pomiędzy warstwami aplikacji i urządzeniami komunikującymi się z serwerem.



Rys. 2. Warstwowy model aplikacji serwerowej

Warstwa ta jest odpowiedzialna za przechowywanie i udostępnianie danych warstwom wyższym. Została ona oparta o wolnodostępny system zarządzania relacyjnymi bazami danych - MySQL. Serwery MySQL są często stosowane w aplikacjach webowych, i obecnie obsługują większość standardu ANSI/ISO SQL, wprowadzając swoje rozszerzenia i elementy języka.

Warstwa przetwarzania danych

Warstwa przetwarzania danych jest odpowiedzialna za gromadzenie, odczyt danych i ich zapis do bazy danych. W tym przypadku możliwe jest bezpośrednie połączenie warstwy klienta oraz warstwy prezentacji, gdyż w obu przypadkach musi być zapewniony dostęp do danych. Warstwa ta jest swoistego rodzaju nakładką dla zarządzania danymi na serwerze MySQL. Wykorzystany został tutaj ORM (ang. Object-Relational Mapping), umożliwiający obiektowe podejście do zarządzania danymi. Wszelkie operacje na bazie danych realizowane są przez ORM, dzięki czemu w znacznym stopniu ułatwia to zarządzanie rozbudowanymi bazami danych. W warstwie przetwarzania znajduje się także logika serwera. Wszystkie dane po wstępnej walidacji musza być odpowiednio przetworzone i gdy zachodzi potrzeba przekazane do warstwy prezentacji.

Warstwa prezentacji

Dostęp do warstwy prezentacji jest możliwy z aplikacji WWW, która dostaje odpowiednio upakowane dane z warstwy niższej. Warto tutaj wspomnieć o wzorcu MVC (ang. Model View Controller), który wspiera framework Symfony 2. Dzięki takiemu rozwiązaniu możliwe jest rozgraniczenie modelu, logiki oraz prezentacji danych, co pozwala na pewną hermetyzację modelu, gdyż logika biznesowa jest odizolowana od wszelkich technologii widoku czy protokołów obsługi żądań wysyłanych przez użytkowników.

Prezentacją danych zajmuje się silnik TWIG oraz jQuery. Wszelkie czynności jakie użytkownik podejmuje są realizowane z wykorzystaniem AJAX (ang. Asynchronous Javascript and XML), dzięki któremu użytkownik może realizować polecenia asynchronicznie, wykonując jednocześnie kilka czynności w serwisie.

Warstwa klienta

Warstwa klienta może być bezpośrednio powiązana z warstwą przetwarzania danych lub warstwą prezentacji [7]. Odpowiada ona interakcję z użytkownikiem, czyli wykonywanie poleceń, które umożliwiają pośrednią komunikacje z bazą danych poprzez warstwę przetwarzania danych (głównie aplikacja mobilna komunikuje się w ten sposób) lub wykorzystując warstwę prezentacji do wizualizacji istniejących informacji.

1.3. Protokół komunikacyjny – XML

Protokół komunikacji z serwerem został opracowany w oparciu o język XML (ang. Extensible Markup Language). Język ten pozwala na reprezentowanie różnych danych w ustrukturyzowany sposób. Dzięki temu, że jest on także niezależny od platformy, umożliwia łatwą wymianę dokumentów pomiędzy heterogenicznymi systemami. Łatwość zapisu oraz dostępność narzędzi do przetwarzania tego typu dokumentów spowodowała, że XML stał się jednym z najczęściej używanych standardów wymiany danych poprzez Internet[3].

W protokole zastosowano kilka rodzajów wiadomości, które są wymieniane pomiędzy aplikacją mobilną a serwerem. Każda wiadomość jest opakowana w tag <*message*>, ten z kolei obowiązkowo zawiera <*type*>. Na podstawie zawartości znacznika <type> serwer jest w stanie określić jakie wykonać czynności w danym przypadku. Obecnie funkcjonalność systemu z wykorzystaniem protokołu umożliwia:

- Aktywację konta. Każdy użytkownik jest zobowiązany do aktywowania swojego konta, przed uprzednim rozpoczęciem sesji komunikacyjnej. Użytkownik musi być zarejestrowany w systemie aby możliwe było wykonanie aktywacji. Aplikacja kliencka robi to automatycznie po wpisaniu nazwy użytkownika i hasła. Na serwer zostaje wysłana wiadomość typu *confirmationRequest*, a po poprawnej odpowiedzi *confirmationEndRequest*. Po wykonaniu tych żądań jeśli serwer nie zwrócił żadnego błędu użytkownik jest informowany o poprawnym aktywowaniu konta. Po aktywacji konta użytkownik odpytuje serwer o dane konfiguracyjne dla rejestratorów oraz dodatkowe informacje dla użytkownika.
- Nawiązanie sesji komunikacyjnej i wymiana danych. Po poprawnej aktywacji konta użytkownik posiada niezbędne dane do rozpoczęcia sesji komunikacyjnej. Każda sesja komunikacyjna posiada swój unikalny identyfikator sesji, który jest po każdym zakończeniu sesji generowany przez serwer i przekazywany klientowi, jako identyfikator do kolejnych połączeń. Przy rozpoczęciu sesji komunikacyjnej z serwerem aplikacja mobilna wysyła żądanie typu *sessionRequest*. Inicjuje to cały proces wymiany danych z serwerem, rozpoczynając od pobrania informacji dodatkowych dla użytkownika i konfiguracji rejestratorów. Urządzenie mobilne mając podłączonych kilka rejestratorów, posiada identyfikatory danych urządzeń, które są zarejestrowane w systemie i przypisane do danego pacjenta. Umożliwia to pobranie nastaw np. minimalnej oraz maksymalnej wartości pomiaru, częstości wykonywania pomiarów przez rejestratory[6]. Na podstawie tych informacji pacjent jest informowany na bieżąco o ewentualnych nieprawidłowościach. Wszystkie pomiary są zapisywane w lokalnej bazie danych urządzenia mobilnego. Dodatkowo po skończeniu każdej sesji komunikacyjnej aplikacja dostaje informacje o czasie kolejnego połączenia. Pozwala to na inteligentne rozładowywanie ilości połączeń w jednym czasie. Dane przesyłane przez aplikację odpowiadają identyfikatorom rejestratorów i są powiązywane z danym pacjentem. W razie niepowodzenia podczas sesji komunikacyjnej, serwer przerywa komunikację i rozpoczyna od nowa cały proces wymiany danych.

Wszystkie typy wiadomości muszą przejść wstępne przetwarzanie przez serwer, aby uniemożliwić dalsze operacje na błędnych danych. W znacznym stopniu zmniejsza to wykorzystanie zasobów serwera gdyż, już na wstępie można określić przydatność odebranych wiadomości. Istnieje kilka

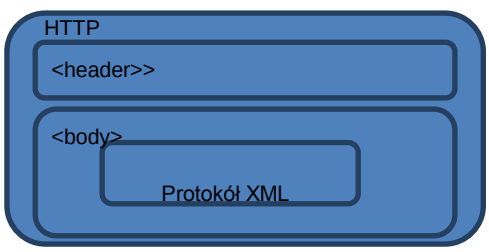
znanych mechanizmów definiowania dokumentów XML, które pozwalają na walidacje zawartości dokumentu. Najbardziej znanymi standardami są DTD (ang. Document Type Definition) oraz XML Schema. Obecnie ze względu na małe możliwości DTD, jest on zastępowany przez XML Schema.

Zastosowanie XML Schema w serwerze w znacznym stopniu ułatwia walidację dokumentów. Sprowadza się to do zdefiniowania struktury dokumentu dla każdego typu wiadomości, zgodnie ze standardem. Parser na podstawie takiego szablonu może określić poprawność struktury oraz przesyłanych danych.

1.4. Komunikacja klient – serwer

Aplikacja kliencka do komunikacji z serwerem jako protokół nadrzędny wykorzystuje protokół HTTP (ang. Hypertext Transfer Protocol), który umożliwia znormalizowany sposób komunikacji klient – serwer. Wiadomości są w postaci tekstowej i mogą zawierać dowolną treść. Lecz HTTP jest tylko protokołem transportowym umożliwiającym właściwą formę komunikacji z serwerem.

Rysunek 3 przedstawia schematyczny pogląd na zawartość wiadomości HTTP. Obsługa błędów związanych z komunikacją czy też informacja o poprawności realizowanych żądań jest wykonywana przez warstwę wyższą, czyli protokół HTTP używając standardowych kodów wiadomości. Pobieranie zasobów i wysyłanie danych na serwer wykorzystuje metody GET i POST. Właściwa zawartość protokołu komunikacyjnego znajduje się w ciele wiadomości HTTP. Uznaje się, że poprawnie wykonane żądanie zawsze kończy się odpowiedzią z serwera w postaci kodu 200. Lecz odpowiedź 200 można uzyskać także w przypadku błędu logicznego. Obsługa błędów logicznych czy też związanych z nieodpowiednim formatem wiadomości przesyłanych przez aplikację mobilną, należy do protokołu XML[8].



Rys. 3. Budowa wiadomości http z uwzględnieniem wewnętrznego protokołu komunikacyjnego

1.5. Bezpieczeństwo

Bezpieczeństwo systemów IT w obecnym czasie jest jednym z najczęściej poruszanych tematów przy tworzeniu tego typu aplikacji[5]. Ważne jest, aby poufne dane na temat użytkowników były zabezpieczone przed niepowołanym dostępem. Dlatego też niezbędna jest odpowiednia analiza ryzyk oraz zaprojektowanie systemu, który będzie w stanie zapewnić określone wymogi.

Istnieje wiele rodzajów ataków jakie mogą być przeprowadzone na serwer, aby uniemożliwić jego działanie bądź dostać się do poufnych informacji. Duża ilość tego typu działań jest niezdefiniowana i poprzez to nie można także określić metod obrony. W opisywanym systemie skupiono się na najbardziej popularnych atakach, na które należałoby zwrócić szczególną uwagę. Do tej grupy z pewnością należy rodzina ataków typu DoS (ang. denial-of-servie), który w większości przypadków sprowadza się do generowania sztucznego ruchu sieciowego, co powoduje czasowe lub trwałe zawieszenie usług serwera lub nawet całkowite zawieszenie serwera podłączonego do sieci Internet. Obrona przed tego typu atakami sprowadza się do odpowiedniej konfiguracji sprzętu sieciowego, który pozwoli na filtrowanie przesyłanych pakietów oraz oprogramowania wykrywającego działania odbiegające od normy.

Warto zwrócić uwagę także na atak SQL Injection, który zazwyczaj opiera się na filtrowaniu znaków ucieczki z danych wejściowych, co pozwala na przekazanie dodatkowych parametrów do zapytania. Problem ten rozwiązuje zastosowanie dodatkowego mechanizmu zarządzającego bazą danych, czyli ORM. Nakłada on pewien poziom abstrakcji na operacje na bazie danych, gdyż pozwala na budowanie dynamicznych zapytań bez konieczności ręcznego ich tworzenia.

2. Podsumowanie i wnioski

Testy aplikacji serwerowej polegały na ustanowieniu sesji komunikacyjnej z aplikacją mobilną odebraniu danych, przeanalizowaniu i zapisaniu w bazie danych. Wzięte zostały pod uwagę różne okoliczności podczas komunikacji tj. tworzenie popranej sesji komunikacyjnej, zerwanie połączenia podczas sesji, generowanie dużych ilości błędnych i poprawnych zapytań. Działania zostały poprawnie zweryfikowane przez serwer. W przypadku zerwania połączenia następowała nowa sesja komunikacyjna i czynności powtarzane aż do stwierdzenia poprawnej sesji. Natomiast jeśli chodzi o generowanie dużej ilości ruchu nie odnotowano znaczących spadków wydajności.

Ta praca powstała przy wsparciu z grantu 2011-2013 numer N N518 284940 dla Nauki Polskiej.

Literatura

[1] Frączkowski K.: Systemy informacyjne oraz usługi w ochronie zdrowia oparte na technologiach SOA (Service Oriented Architecture), Acta Bio-Optica et Informatica Medica 1/2010, vol. 16.
[2] Kasiak. K., Surtel W., Maciejewski R.: Telemedycyna w sytuacjach kryzysowych. Ostry dyżur, 4/2012
[3] Litwińczuk K., Surtel W.: Model bazy danych w telemedycznej obsłudze pacjenta, IAPGOS, 2012, nr 4a, 39-41.
[4] Maciejewski M., Małecka-Massalska T., Surtel W.: New Type of Sensor for Heart Rhythm Monitoring, W: New Trends in Audio and Video, Signal Processing: Algorithms, Architectures, Arrangements and Applications, Łódź, 2012.
[5] Nałęcz M., Kącki E., Kulikowski J. L., Nowakowski A. i Waniewski E.: Biocybernetyka i inżynieria biomedyczna 2000. Tom 7: Systemy komputerowe i teleinformatyczne w służbie zdrowia, Warszawa, Akademicka oficyna wydawnicza Exit, 2002.
[6] Surtel W.: Remote measurements of selected vital functions, W:7th International Conference New Electrical and Electronic Technologies and their Industrial Implementation, Zakopane, 2011.
[7] Wandschneider M.: PHP i MySQL. Tworzenie aplikacji WWW, Helion, Gliwice, 2006.
[8] Xiao Y., Chen H.: Mobile Telemedicine: A Computing and Networking Perspective. CRC Press Auerbach Publications, Floryda, 2008.

Dr inż. Wojciech Surtel
e-mail: w.surtel@pollub.pl

Ukończył studia na Wydziale Elektrycznym Politechniki Lubelskiej. W latach 1989 – 2011 pracownik naukowo–dydaktyczny (asystent, adiunkt) w Katedrze Elektroniki Politechniki Lubelskiej. Tytuł doktora uzyskał w 1999 r. na Wydziale Elektrycznym PL. Temat rozprawy: Cyfrowe przetwarzanie sygnału pomiarowego w wybranych przypadkach dynamicznego ważenia masy. Od roku 2011 starszy wykładowca w Instytucie Elektroniki i Technik Informacyjnych.
Obszar zainteresowań naukowych i dydaktycznych to: telemedycyna i systemy mobilne.

Mgr inż. Marcin Maciejewski
e-mail: m.maciejewski@pollub.pl

Doktorant na wydziale Elektrotechniki i Informatyki Politechniki Lubelskiej. Zajmuje się przetwarzaniem danych w medycynie, urządzeniami do zastosowania w telemedycynie, przetwarzaniem obrazu i systemami mikroprocesorowymi. Autor i współautor prac z dziedziny bioinformatyki, telemedycyny, symulacji komputerowej w biochemii oraz przetwarzania obrazów.

Inż. Rafał Różalski
e-mail: michal.cieslar@pollub.edu.pl

Student II roku studiów stacjonarnych II stopnia na Wydziale Elektrotechniki i Informatyki Politechniki Lubelskiej na kierunku Informatyka, specjalności Technologie wytwarzania oprogramowania

otrzymano/received: 04.05.2013 *przyjęto do druku/accepted: 28.10.2013*